02212
RSW

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００２年１２月１８日

出願番号
Application Number: 特願２００２－３６６３４５

[ST.10/C]: [ＪＰ２００２－３６６３４５]

出願人
Applicant(s): インターナショナル・ビジネス・マシーンズ・コーポレーション

２００３年　６月　９日

特許庁長官
Commissioner,
Japan Patent Office 太田信一郎

出証番号　出証特２００３－３０４４８２７

【書類名】　特許願

【整理番号】　JP9020212

【あて先】　特許庁長官殿

【国際特許分類】　G06F 11/28

【発明者】

【住所又は居所】　神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　大和事業所内

【氏名】　稲垣　巌

【発明者】

【住所又は居所】　神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　大和事業所内

【氏名】　古澤　修

【特許出願人】

【識別番号】　390009531

【氏名又は名称】　インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】

【識別番号】　100086243

【弁理士】

【氏名又は名称】　坂口　博

【代理人】

【識別番号】　100091568

【弁理士】

【氏名又は名称】　市位　嘉宏

【代理人】

【識別番号】　100108501

【弁理士】

【氏名又は名称】　上野　剛史

【復代理人】

【識別番号】　100085408

【弁理士】

【氏名又は名称】　山崎　隆

【手数料の表示】

【予納台帳番号】　117560

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9706050

【包括委任状番号】　9704733

【包括委任状番号】　0207860

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　情報処理装置、トレース処理方法、プログラム及び記録媒体

【特許請求の範囲】

【請求項１】　処理状況に関する情報を収集するトレース処理が含まれるルーチンをマルチスレッドで実行可能な情報処理装置において、

起動中のスレッド毎に、実行中のルーチンの登録を行うルーチン登録手段と、

実行中のルーチンにおけるトレース処理のレベルを前記登録情報に基づいて判定するレベル判定手段とを具備することを特徴とする情報処理装置。

【請求項２】　前記レベル判定手段は、前記実行中のルーチンにおけるトレース処理のレベル判定を、そのルーチンと同一又は所定の関係にあるルーチンの登録がなされているか否かに基づいて行うものであることを特徴とする請求項１に記載の情報処理装置。

【請求項３】　前記所定の関係を相互に有するルーチンを規定するテーブルを有することを特徴とする請求項２に記載の情報処理装置。

【請求項４】　前記テーブルの設定又は変更を行うための入力を受け入れ、前記テーブルの設定又は変更を行う手段を有することを特徴とする請求項３に記載の情報処理装置。

【請求項５】　前記所定の関係にあるルーチンは、共有のリソースにアクセスする関係にあるルーチンであることを特徴とする請求項２に記載の情報処理装置。

【請求項６】　前記レベル判定手段は、前記実行中のルーチンにおけるトレース処理のレベル判定に際し、そのルーチンと同一又は所定の関係にあるルーチンの登録がなされている場合、トレース処理のレベルはトレース処理の実行が制限なく許容されるレベルである旨の判定を行うものであることを特徴とする請求項１に記載の情報処理装置。

【請求項７】　前記レベル判定手段は、前記実行中のルーチンにおけるトレース処理のレベル判定に際し、そのルーチンと同一又は所定の関係にあるルーチンの登録がなされていない場合、トレース処理のレベルはトレース処理の実行がまったく許容されないレベルである旨の判定を行うものであることを特徴とする

請求項１に記載の情報処理装置。

【請求項８】　前記レベル判定手段は、前記登録が行われる際に、登録対象ルーチンと同一又は所定の関係にあるルーチンの登録がなされている場合、その旨を示す同時実行情報を、双方の登録情報に対して付加するものであることを特徴とする請求項１に記載の情報処理装置。

【請求項９】　前記レベル判定手段は、前記同時実行情報の付加を行う際に、前記双方の登録情報に対し、前記トレース処理のレベルを示すトレース・レベル情報を付加するものであることを特徴とする請求項８に記載の情報処理装置。

【請求項１０】　前記トレース処理が含まれるルーチンを含むサーバ・プログラムと、クライアント・プログラムからの前記ルーチンの呼び出しに応じて前記ルーチンの実行を行うアプリケーション実行制御部と、前記アプリケーション実行制御部からの依頼に応じてトレース処理を実行するトレース実行部とを備え、前記トレース実行部はトレース処理の実行に際し、そのトレース処理のレベルを前記レベル判定手段に問い合わせるものであることを特徴とする請求項１に記載の情報処理装置。

【請求項１１】　前記トレース処理が含まれるルーチンを使用するプログラムと、前記トレース処理が含まれるルーチンが格納された共有ライブラリと、前記プログラムで使用されるルーチンからの依頼に応じてトレース処理を実行するトレース実行部とを備え、前記トレース実行部はトレース処理の実行に際し、そのトレース処理のレベルを前記レベル判定手段に問い合わせるものであることを特徴とする請求項１に記載の情報処理装置。

【請求項１２】　処理状況に関する情報を収集するトレース処理が含まれるルーチンをマルチスレッドで実行可能な情報処理装置におけるトレース処理方法であって、

起動中のスレッド毎に、実行中のルーチンの登録を前記情報処理装置が行うルーチン登録工程と、

実行中のルーチンにおけるトレース処理のレベルを前記登録情報に基づいて前記情報処理装置が判定するレベル判定工程とを具備することを特徴とするトレース処理方法。

【請求項１３】　前記レベル判定工程では、前記実行中のルーチンにおけるトレース処理のレベル判定を、そのルーチンと同一又は所定の関係にあるルーチンの登録がなされているか否かに基づいて行うことを特徴とする請求項１２に記載のトレース処理方法。

【請求項１４】　前記トレースのレベル判定に際し、前記所定の関係を相互に有するルーチンを規定するテーブルを参照することを特徴とする請求項１３に記載のトレース処理方法。

【請求項１５】　前記情報処理装置が、前記テーブルの設定又は変更を行うための入力を受け入れ、前記テーブルの設定又は変更を行う工程を有することを特徴とする請求項１４に記載のトレース処理方法。

【請求項１６】　前記所定の関係にあるルーチンは、共有のリソースにアクセスする関係にあるルーチンであることを特徴とする請求項１３に記載のトレース処理方法。

【請求項１７】　前記レベル判定工程では、前記実行中のルーチンにおけるトレース処理のレベル判定に際し、そのルーチンと同一又は所定の関係にあるルーチンの登録がなされている場合、トレース処理のレベルはトレース処理の実行が制限なく許容されるレベルである旨の判定を行うことを特徴とする請求項１２に記載のトレース処理方法。

【請求項１８】　前記レベル判定工程では、前記実行中のルーチンにおけるトレース処理のレベル判定に際し、そのルーチンと同一又は所定の関係にあるルーチンの登録がなされていない場合、トレース処理のレベルはトレース処理の実行がまったく許容されないレベルである旨の判定を行うものであることを特徴とする請求項１２に記載のトレース処理方法。

【請求項１９】　前記レベル判定工程では、前記登録が行われる際に、登録対象ルーチンと同一又は所定の関係にあるルーチンの登録がなされている場合、その旨を示す同時実行情報を、双方の登録情報に対して付加することを特徴とする請求項１２に記載のトレース処理方法。

【請求項２０】　前記レベル判定工程では、前記同時実行情報の付加を行う際に、前記双方の登録情報に対し、前記トレース処理のレベルを示すトレース・

レベル情報を付加することを特徴とする請求項１９に記載のトレース処理方法。

【請求項２１】　前記トレース処理が含まれるルーチンを含むサーバ・プログラムを前記情報処理装置が実行する工程と、クライアント・プログラムからの呼び出しに応じて前記ルーチンの実行を前記情報処理装置が行うアプリケーション実行制御工程と、前記アプリケーション実行制御工程で発生したトレース処理を前記情報処理装置が実行するトレース実行工程とを備え、前記トレース実行工程ではトレース処理の実行に際し、前記レベル判定工程における判定結果を参照することを特徴とする請求項１２に記載のトレース処理方法。

【請求項２２】　前記トレース処理が含まれるルーチンを使用するプログラムを前記情報処理装置が実行する工程と、共有ライブラリに格納された前記トレース処理が含まれるルーチンを前記情報処理装置が実行する工程と、前記ルーチン実行工程におけるトレース処理を前記情報処理装置が実行するトレース実行工程とを備え、前記トレース実行工程ではトレース処理の実行に際し、前記レベル判定工程における判定結果を参照することを特徴とする請求項１２に記載のトレース処理方法。

【請求項２３】　コンピュータを請求項１～１１のいずれかの情報処理装置及びそれを構成する各手段として機能させることを特徴とするプログラム。

【請求項２４】　コンピュータを、

該コンピュータにおいて起動中のスレッドで実行中のルーチンを登録する旨の依頼に応じ、スレッド毎にその登録を行うルーチン登録手段、及び

実行中のルーチンにおけるトレース処理のレベルの問合せに応じ、前記登録情報に基づき、前記問合せに対する回答を行う回答手段として機能させることを特徴とするプログラム。

【請求項２５】　請求項２３又は２４のプログラムを記録したことを特徴とするコンピュータ読取り可能な記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、トレース処理が含まれるルーチンをマルチスレッドで実行可能な情

報処理装置、この情報処理装置におけるトレース処理方法、コンピュータを前記情報処理装置として機能させるプログラム、及びこのプログラムを記録したコンピュータ読取り可能な記録媒体に関する。

【０００２】

【従来の技術】

従来、プログラムに障害が発生した場合にその原因を解析し、特定するため、プログラムの実行状況を標準出力やファイル等に時系列で出力するトレースという手法が知られている。しかしながら、トレースを用いてより多くの情報を取得しようとすると、本来の業務ロジックの処理以外に、プログラムの実行状況を記録するための余分な処理が必要となるため、プログラム全体としてのパフォーマンスの低下を招く。また、適切な範囲で情報の収集を行わないと、大量のデータが出力されることになり、システム・リソースを圧迫する要因となりうる。

【０００３】

そこで、トレースを必要最小限の範囲で実施することによって、パフォーマンスの低下やシステム・リソースの圧迫を抑えることを目的として従来、次のような工夫が行われている。たとえば、収集すべき情報を重要度に応じて分類し、必要とする情報の重要度をユーザが指定することによって重要度の低い情報が取得されるのを防止するものや、特定のスレッドに対するトレース機能を設け、特定のスレッド上で実行される処理についてのみ情報の収集を行うものが知られている。

【０００４】

また、スレッドを排他実行スレッドと、他のスレッドとに分類し、排他実行スレッドについては同時実行させないようにしてスレッドの実行順序を制御しつつ情報収集を行うことにより、デバッグ効率の向上を図ったデバッグ装置も知られている（たとえば特許文献１参照）。また、通常のインタプリタと、トレース用のインタプリタとを設け、これらを切り替えて使用することにより、トレース機能の実行可否を制御できるようにした技術も知られている（たとえば特許文献２参照）。

【０００５】

【特許文献１】

特開平１１－３３８７３３号公報

【特許文献２】

特開２００２－５５８４８号公報

【０００６】

【発明が解決しようとする課題】

しかしながら、これら従来のトレース技術によれば、プログラムの実行前又は実行中にユーザによってなされる設定又は設定変更に従って予め特定されるスレッドやプログラム部分等についてのみトレースの可否を制御し得るだけである。つまり、実際の運用下における種々の実行要求に応じて動的に変化するスレッドの生成状況に対応した情報収集を行うことができない。したがって、従来のトレース技術をマルチスレッド環境下でのみ発生する障害の分析に適用すると、障害の発生条件に合致しない場合、たとえばシングル・スレッドでプログラムが実行されるような場合においてもトレース処理による情報の収集が行われ、収集された情報には障害の発生原因とは無関係のデータが多く含まれることになる。このためパフォーマンスの低下やシステム・リソースの圧迫に加え、トレース出力の分析に大きな労力が必要となる。

【０００７】

また、マルチスレッド・プログラムにおいて、複数のスレッドが同時に特定の処理を実行することが原因となって発生する障害は、各スレッドの実行状況などのタイミングに依存する。このため、単一のスレッドで発生する障害に比べて問題の原因を特定することが困難である。

【０００８】

さらに、マルチスレッド環境下でのみ発生する障害を分析する目的でトレースを使用する場合、従来のトレース技術によれば、収集する情報には障害の発生原因とは無関係のデータが多く含まれ、多くの余分な処理が発生するので、実際の運用環境においてはトレースの取得が許されない場合がある。

【０００９】

本発明の目的は、このような従来技術の問題点に鑑み、複数のスレッドにおい

て共通の資源をアクセスすることが原因で発生する問題の解析を、プログラムの実行環境に大きな影響を与えることなく、容易に行うことができるようなトレース技術を提供することにある。

【００１０】

【課題を解決するための手段】

この目的を達成するため、本発明に係るトレース処理方法は、処理状況に関する情報を収集するトレース処理が含まれるルーチンをマルチスレッドで実行可能な情報処理装置におけるトレース処理方法であって、起動中のスレッド毎に、実行中のルーチンの登録を前記情報処理装置が行うルーチン登録工程と、実行中のルーチンにおけるトレース処理のレベルを前記登録情報に基づいて前記情報処理装置が判定するレベル判定工程とを具備することを特徴とする。

【００１１】

ここで、トレース処理としては、たとえば所定のメッセージや所定の変数の内容をログ・ファイル等に出力する処理が該当する。ルーチンとは何らかの処理を行う一連のプログラムコードであり、たとえば、関数やメソッドが該当する。トレース処理のレベルの判定としては、たとえばトレース処理を行うか否かの判定や、トレース処理を収集可能なすべての情報について行うか、一部の情報については省略して行うか、又はまったく行わないかの判定が該当する。実行中のルーチンの登録とは、実行開始時に登録を行い、実行終了時に登録を抹消することを意味する。

【００１２】

この構成において、ルーチン登録工程では、ルーチンの実行のためにスレッドが起動するときにそのルーチンの登録を行い、実行が終了してスレッドが消滅するときにルーチンの登録を抹消する。これにより、実行中のルーチンの登録がスレッド毎に行われる。この実行中のルーチンの登録情報に基づき、レベル判定工程では、実行中のルーチンにおけるトレース処理のレベルを判定する。その際、たとえば、同一ルーチンが異なるスレッドで実行中であればトレース処理は許容され、さもなければ許容されないレベルであると判定することができるが、このような判定結果は、登録情報の経時的変化に応じて刻々変化する。つまり、レベ

ル判定工程では、実行中のルーチンにおけるトレース処理の可否や程度が、どのようなルーチンが他のスレッドで実行中であるかに応じ、動的に判定されることになる。この判定結果に基づき、トレース処理の可否やレベルを動的に制御することができる。

【００１３】

本発明に係るトレース処理方法において、前記レベル判定工程では、実行中のルーチンにおけるトレース処理のレベル判定を、そのルーチンと同一又は所定の関係にあるルーチンの登録がなされているか否かに基づいて行うことができる。この場合、前記所定の関係を相互に有するルーチンを規定するテーブルを参照するようにしてもよい。さらに、情報処理装置がこのテーブルの設定又は変更を行うための入力を受け入れ、テーブルの設定又は変更を行うようにしてもよい。前記所定の関係にあるルーチンとしてはたとえば、共有のリソースにアクセスする関係にあるルーチンや、同時に実行すると、何らかの原因により問題が生じ得る関係にあるルーチンが該当する。

【００１４】

また、本発明に係るトレース処理方法において、レベル判定工程としてはたとえば、実行中のルーチンにおけるトレース処理のレベル判定に際し、そのルーチンと同一又は所定の関係にあるルーチンの登録がなされており又はなされていない場合、トレース処理のレベルはトレース処理の実行が制限なく許容され又はまったく許容されないレベルである旨の判定を行うものが該当する。あるいは、ルーチンの登録が行われる際に、登録対象ルーチンと同一又は所定の関係にあるルーチンの登録がなされている場合、その旨を示す同時実行情報を、双方の登録情報に対して付加するものが該当する。同時実行情報の付加を行う際に、双方の登録情報に対し、トレース処理のレベルを示すトレース・レベル情報を付加するようにしてもよい。

【００１５】

また、本発明に係るトレース処理方法は、トレース処理が含まれるルーチンを含むサーバ・プログラムを情報処理装置が実行する工程と、クライアント・プログラムからの呼び出しに応じて前記ルーチンの実行を情報処理装置が行うアプリ

ケーション実行制御工程と、アプリケーション実行制御工程で発生したトレース処理を情報処理装置が実行するトレース実行工程とを備えてもよい。あるいは、トレース処理が含まれるルーチンを使用するプログラムを情報処理装置が実行する工程と、共有ライブラリに格納された前記トレース処理が含まれるルーチンを情報処理装置が実行する工程と、前記ルーチン実施工程におけるトレース処理を情報処理装置が実行するトレース実行工程とを備えてもよい。これらの場合、トレース実行工程では、トレース処理の実行に際し、レベル判定工程における判定結果を参照することができる。

【００１６】

本発明に係る情報処理装置は、上述の本発明に係るトレース処理方法を構成する各工程を行う手段によって構成される。

【００１７】

本発明に係るプログラムは、本発明に係る情報処理装置及びそれを構成する各手段としてコンピュータを機能させるものであることを特徴とする。

【００１８】

本発明に係る記録媒体は、本発明に係るプログラムを記録したコンピュータ読取り可能な記録媒体であることを特徴とする。

【００１９】

【発明の実施の形態】

【００２０】

図１は本発明を適用し得るＯＲＢ（Object Request Broker）による通信基盤を基礎とするクライアント・サーバ・システムを示す。ＯＲＢはクライアントとサーバ・オブジェクトとの間の接続を確立し、クライアントがサーバ・オブジェクトのメソッドを呼び出すことを可能にするソフトウェアである。このシステムにおいて稼動するクライアント・サーバ・アプリケーションはサーバ・プログラム１及びクライアント・マシーン２におけるクライアント・プログラムを有する。サーバ・プログラム１が定義するオブジェクトのメソッドはアプリケーション実行制御部３により実行される。サーバ・プログラム１ではたとえばクラスＡに属するインスタンスａ及びｂが定義されており、各インスタンスはトレース処理

が含まれたメソッドＸ及びメソッドＹを備えている。アプリケーション実行制御部３は、たとえばＪａｖａ（登録商標）仮想マシンに相当するものであり、プラットフォームに依存しないバイトコードを実行することによって、サーバのＯＳ（オペレーティング・システム）の如何に関係なく、メソッドの実行を可能にしている。

【００２１】

クライアント・プログラムがメソッドＸを呼び出すと、アプリケーション実行制御部３は、ＯＳが生成する単一のスレッドにおいて、メソッドＸを構成するバイトコードを順次解釈して実行する。異なるクライアント・マシーン２における各クライアント・プログラムがメソッドＸを同時に呼び出すと、異なるアプリケーション実行制御部３が、クラスＡのインスタンスａ及びｂのメソッドＸを、異なるスレッドにおいて並行して実行する。

【００２２】

このようなシステムにおいては、従来のトレース手法によれば、グローバル変数等の静的に設定されたトレース制御情報を参照し、全スレッドにおいてそれぞれトレース処理の必要性の有無を判断していた。これに対し、本発明の第１の実施形態においては図２に示すように、図１のシステムに対し、トレース制御部（Ｔｒａｃｅ　Ｍａｎａｇｅｒ）２１及びスレッド処理多重度管理テーブル２２を導入し、トレース実行部（ＴｒａｃｅＥｘｅｃｔｉｏｎｅｒ）２３によるログ・ファイル２４へのトレース情報の出力を、起動中のスレッドで実行されているメソッドが同一であるか否かに応じ、動的に制御するようにしている。

【００２３】

スレッド処理多重度管理テーブル２２は、起動中のスレッド毎に、実行中のメソッドを登録するためのテーブルである。テーブルを構成する各レコードは、起動中のスレッドを識別するためのスレッド識別子、対応スレッドで実行中であるメソッドを特定するクラス名及びメソッド名、並びにトレース・レベルの各項目を有する。トレース・レベルの項目には、対応スレッドで実行中のメソッドにおけるトレース処理の実行可否を示すフラグがセットされる。フラグとしては、トレース処理の実行を許容する旨を示す”５”及びトレース処理の実行を禁止する旨

を示す”０”が用いられる。

【００２４】

トレース制御部２１は、アプリケーション実行制御部３からの要求に応じ、生成されたスレッドの管理テーブル２２への登録を行うメソッド”ａｄｄＴｈｒｅａｄＩｎｆｏ（）”、及び消去されたスレッドの登録を管理テーブル２２から抹消するメソッド”ｒｅｍｏｖｅＴｈｒｅａｄＩｎｆｏ（）”を備えている。またトレース実行部２３からの、スレッド識別子を特定した問合せに応じ、そのスレッドにおけるトレース処理の可否について、管理テーブル２２を参照してチェックし、回答を行うメソッド”ｃｈｅｃｋＴｒａｃｅ（）”を備える。トレース実行部２３はアプリケーション実行制御部３の要求に応じ、トレース処理の可否をトレース制御部２１に問い合せ、回答結果に応じてトレース処理を実行し、又は省略するメソッド”ｓｅｔＬｏｇＥｖｅｎｔ（）”を備える。

【００２５】

この構成において、クライアント・プログラムがサーバ・プログラム１のメソッドを呼び出すと、サーバではＯＳによりスレッドが生成され、対応するアプリケーション実行制御部３に対し、呼び出されたメソッドの実行要求がなされる（図中の①の処理）。

【００２６】

この実行要求を受けると、アプリケーション実行制御部３は、呼び出されたメソッドを実際に実行する前に、トレース制御部２１に対し、生成されたスレッドに関してスレッド処理多重度管理テーブル２２への登録を要求するとともに、生成されたスレッドの識別子、呼び出されたメソッドのクラス名及びメソッド名を通知する（Call addThreadInfo()；処理②）。

【００２７】

この要求に応じ、トレース制御部２１は、通知されたスレッドの識別子並びにクラス名及びメソッド名をスレッド処理多重度管理テーブル２２に登録する（addThreadInfo()；処理③）。その際、トレース制御部２１は、登録するメソッドとクラス名及びメソッド名が同一であるメソッドが管理テーブル２２に登録済みであるかどうかを調べ、登録済みであることが判明した場合は、それら同一メソ

ッドの登録レコードにおけるトレース・レベルを”５”に設定する（処理④）。

【００２８】

たとえば、スレッド処理多重度管理テーブル２２が図２に示すような状態にあり、登録されるスレッドの識別子が”５００”、メソッドのクラス名及びメソッド名が”ｃｌａｓｓ　Ａ”及び”Ｍｅｔｈｏｄ　Ｘ”であるとすれば、これと同一のクラス名及びメソッド名のメソッドが、スレッド識別子が”１００”のものとして登録済みであるため、スレッド識別子が”５００”及び”１００”のレコードにおけるトレース・レベルが”５”にセットされることになる。

【００２９】

一方、アプリケーション実行制御部３は、クライアント・プログラムが呼び出したメソッドを実行しているときにトレース処理たとえばプリント命令”Ｐｒｉｎｔ”Ｔｒａｃｅ２””に遭遇すると、そのトレース処理をトレース実行部２３に依頼する（Print”Trace2”；処理⑤）。この依頼を受けると、トレース実行部２３は、トレース制御部２１に対し、依頼されたトレース処理の実行の可否について問合せを行う（処理⑥）。この問合せには、依頼されたトレース処理に係るスレッドの識別子を伴う。

【００３０】

この問合せに応じ、トレース制御部２１は、スレッド処理多重度管理テーブル２２中の問合せに係るスレッド識別子のレコードにおけるトレース・レベルの値を調べ（checkTrace()；処理⑦）、トレース処理の可否をトレース実行部２３に通知する（処理⑥）。この通知に基づきトレース実行部２３は、依頼されたトレース処理を実行し（setLogEvent()；処理⑧）、又は省略する。たとえば、依頼されたトレース処理が、識別子が”１００”のスレッドにおけるトレース処理であるとすれば、スレッド処理多重度管理テーブル２２における該当レコードのトレース・レベルは”５”であるため、トレース処理が実行されることになる。

【００３１】

他方、クライアント・プログラムから呼び出されているずれかのメソッドの実行が終了し、対応するスレッドが消去されるとき、対応するアプリケーション実行制御部３は、消去されるスレッドに関する登録の抹消をトレース制御部２１に

依頼する（Call removeThreadInfo()；処理⑨）。この依頼を受けると、トレース制御部２１は、登録抹消を依頼されたスレッドに関する登録をスレッド処理多重度管理テーブル２２から抹消する（removeThreadInfo()；処理(10)）。その際、削除するレコードとクラス名及びメソッド名が同一であるレコードについては、さらにクラス名及びメソッド名が同一である他のレコードが存在しない限り、トレース・レベルを”０”とする。たとえば、図２の管理テーブル２２の状態においてスレッド識別子が”１００”であるレコードが削除されるとき、そのクラス名”ｃｌａｓｓ　Ａ”及びメソッド名”Ｍｅｔｈｏｄ　Ｘ”が同一であるスレッド識別子が”５００”のレコードのトレース・レベルは”０”に変更されることになる。

【００３２】

図３～図５は上述のトレース制御部２１の処理を示す。トレース制御部２１はアプリケーション実行制御部３からの生成されたスレッドに関する登録依頼又は消去されるスレッドに関する登録抹消依頼を受けた場合は図３の登録処理（addThreadInfo()）又は図４の登録抹消処理（removeThreadInfo()）を行い、トレース実行部２３からのトレース処理の可否に関する問合せがあった場合には図５の実行可否チェック処理（checkTrace()）を行う。

【００３３】

登録に際しては、図３に示すように、スレッド処理多重度管理テーブル２２に対し、図２に示される形式の１レコードを追加するが（ステップ４１）、その際、同一メソッドに関する登録が既に行われている場合は双方のレコードにおけるトレース・レベルを”５”にセットし、さもなければ追加レコードにおけるトレース・レベルを”０”にセットする（ステップ４２～４４）。登録抹消に際しては図４に示すように、対応レコードを削除するとともに（ステップ５１）、削除レコードと同一のメソッドに関するレコードが存在し、かつそのようなレコードが１つのみである場合はそのトレース・レベルを”０”にセットする（ステップ５２、５３）。トレース処理の実行可否のチェックに際しては、図５に示すように、そのトレース処理に係るスレッドの識別子をキーにして対応レコードのトレース・レベルをチェックし、その結果をトレース実行部２３に通知する（ステップ６１、６２）。

【００３４】

このようにしてトレース制御部２１は常に、実行中のスレッドのみがすべて登録され、かつクラス名及びメソッド名が同一であるスレッドについては、トレース・レベルが”５”となるようにスレッド処理多重度管理テーブル２２の登録内容を維持し、管理するとともに、スレッド処理多重度管理テーブル２２に基づいて、各スレッドにおけるトレース処理の実行可否のチェックを行う。

【００３５】

本実施形態によれば、トレース処理が含まれるメソッドをマルチスレッドで実行可能なサーバにおいて、起動中のスレッド毎に、実行中のメソッドの登録を行い、実行中のメソッドにおけるトレース処理の可否を登録情報に基づいて判定するようにしたため、判定時点で起動中の各スレッドにおいて実行されているメソッドに応じ、動的にトレース処理の可否の判定を行うことができる。つまり、共有リソースにアクセスすることによる問題が生じる可能性は、どのようなメソッドが同時に実行されているかに応じて刻々変化するが、その変化する状況に応じて、問題解決のためのトレース処理を、問題が生じる可能性がある場合に限定して行うことができる。したがって、プログラムの実行環境に大きな影響を与えることなく、不要な情報の収集を排除したトレース処理を行い、問題解決のための解析の容易化を図ることができる。

【００３６】

また、実行中のメソッドにおけるトレース処理の可否の判定を、そのメソッドと同一の関係にあるメソッドの登録がなされているか否かに基づいて行うようにしたため、共有リソースにアクセスすることによる問題が生じる可能性が高い場合に限定してトレース処理を行うことができる。

【００３７】

また、スレッド処理多重度管理テーブル２２への登録が行われる際に、登録対象メソッドと同一の関係にあるメソッドの登録がなされている場合、その旨を示す同時実行情報を”５”として、双方の登録レコードにおけるトレース・レベルの項目に記録するようにしたため、トレース処理の実行可否の判定結果が必要な時には、スレッド処理多重度管理テーブル２２を再度探索することなく直ちに判定

結果を得ることができる。

【００３８】

以上の実施形態では最も単純な条件、すなわち同一メソッドの同時実行が開始された時点でトレース処理の実行可否の条件を更新するようにしているが、次に示す第２の実施形態では、より複雑な条件でトレース処理実行可否の制御を行うようにしている。すなわちトレース条件管理テーブルを導入し、このテーブルにおいてトレース処理の実行可否の条件をより詳細に規定することにより、トレース処理実行可否の制御をより柔軟に行うことを可能にしている。

【００３９】

図６は本発明の第２の実施形態に係るシステムの主要部、及び処理の流れを示す。このシステムでは、第１実施形態におけるトレース制御部２１の代わりにトレース制御部３１を備え、スレッド処理多重度管理テーブル２２のトレース・レベルを、トレース条件管理テーブル３２を参照して設定又は変更するようにしている。また、トレース・レベルとして、トレース処理の許容又は省略の２つのレベルのみではなく、より多くのレベルを設定できるようにしている。この設定レベルに応じて、トレース処理可否の問合せに対する回答が行われることになる。他の点については上述第１実施形態の場合と同様である。

【００４０】

トレース条件管理テーブル３２は複数のトレース・パターンを定義している。テーブル中の各レコードはトレース・パターン、トレース・レベル、及びクラス／メソッドの各項目を有する。トレース・パターンは各トレース・パターンを区別するためのパターン番号、トレース・レベルの項目には対応トレース・パターンにおけるトレース処理の程度を示すレベル番号、クラス／メソッドの項目には対応トレース・パターンが適用されるメソッドを特定するクラス名及びメソッド名が格納される。

【００４１】

各レコードにおけるクラス／メソッドの項目には特定の関係を有する複数のメソッドのクラス名及びメソッド名を記録することができる。この特定関係を有するメソッド同士については、同時に実行されるとき、それらのメソッドにおける

トレース処理について、それらのメソッドを前記特定関係を有するものとして規定しているレコードのトレース・レベルが適用される。また、同一のメソッドが同時に実行されているときには、そのメソッドについて規定されているトレース・レベルが適用される。

【００４２】

前記特定関係を有するメソッドとしては、たとえば共通のリソースにアクセスするメソッドが該当する。図６の例では、トレース・パターン”２”のレコードにおいて、”ｃｌａｓｓ　Ａ：：Ｍｅｔｈｏｄ　Ｙ”及び”ｃｌａｓｓ　Ｂ：：Ｍｅｔｈｏｄ　Ｚ”が前記特定関係を有するメソッドとして規定されている。

【００４３】

図７～図９はそれぞれ、トレース制御部３１による登録処理、登録抹消処理、及び実行可否チェック処理を示す。トレース制御部３１はアプリケーション実行制御部３（図２）からの生成されたスレッドに関する登録依頼又は消去されるスレッドに関する登録抹消依頼を受けた場合は図７の登録処理（addThreadInfo()）又は図８の登録抹消処理（removeThreadInfo()）を行い、トレース実行部２３からのトレース処理の可否に関する問合せがあった場合には図９の実行可否チェック処理（checkTrace()）を行う。

【００４４】

登録処理においては、図７に示すように、まず、そのスレッドが実行するメソッドと同一のメソッドがクラス／メソッドの項目に記録されているレコードをトレース条件管理テーブル３２においてサーチし、対応するトレース・レベルを取得する（ステップ８１）。また、当該レコード中に前記特定関係にある他のメソッドのクラス名及びメソッド名が記録されていればそれも取得する（ステップ８２）（図６中の処理①）。

【００４５】

次に、登録依頼を受けたスレッドの識別子、並びにそのスレッドが実行するメソッドのクラス名及びメソッド名を記録したレコードを追加することにより、スレッド処理多重度管理テーブル２２に対して新たなエントリを加える（ステップ８３）。このとき、追加レコードと同一のクラス名及びメソッド名を有するレコ

ードがスレッド処理多重度管理テーブル２２中に存在するかどうかを判定し（ステップ８４）、存在する場合には、そのレコード及び追加レコード双方のトレース・レベルを前記取得したトレース・レベルに設定する（ステップ８６）。また前記取得したクラス名及びメソッド名と同一のものがスレッド処理多重度管理テーブル２２中に存在するかどうかを判定し（ステップ８４）、存在する場合にはそのレコード及び追加レコード双方のトレース・レベルを、前記取得したトレース・レベルに設定する（ステップ８６）。同一又は取得したクラス名及びメソッド名と同一のものを有するレコードが存在しない場合は追加レコードのトレース・レベルを”０”に設定する（ステップ８７）。これにより、スレッドの登録が終了する。

【００４６】

一方、登録抹消処理においては、図８に示すように、まず、そのスレッドが実行するメソッドと前記特定関係を有するメソッドがトレース条件管理テーブル３２において規定されているかどうかを調べ（ステップ９１）、特定関係を有するメソッドがあれば、そのクラス名及びメソッド名を取得する（ステップ９２）。

【００４７】

次に登録抹消依頼を受けたスレッドのレコードを削除する（ステップ９３）。このとき、削除レコードに係るメソッドと同一又は前記特定関係にあるメソッドのクラス名及びメソッド名を有するレコードがスレッド処理多重度管理テーブル２２中に存在するか否かを判定し（ステップ９４）、存在する場合には、そのようなレコードがさらに他に存在する場合を除き、そのレコードのトレース・レベルを”０”に設定する（ステップ９５）。これにより、スレッドの登録抹消が終了する。

【００４８】

以上のスレッドの登録及び登録抹消の処理を通じて、スレッド処理多重度管理テーブル２２には常に、起動中のスレッドのみが、実行中のメソッドを特定するクラス名及びメソッド名並びにトレース処理の可否及びレベルに関する情報とともに登録されることになる。

【００４９】

実行可否チェック処理においては、図９に示すように、問合せに係るスレッドに対応するトレース・レベルを、スレッド識別子をキーとしてスレッド処理多重度管理テーブル２２から取得し（ステップ１０１）、トレース実行部２３に送る（ステップ１０２）。

【００５０】

トレース実行部２３は、送られたトレース・レベルに応じ、トレース処理を実行する。たとえば、トレース・レベルが”０”であればトレース処理を行わず、それ以外の値であれば、その値に応じて、トレース処理を部分的又は全体的に行ったり、一部省略したりする。

【００５１】

以上の登録及び登録抹消処理並びにトレース実行可否チェック処理の一例を具体的に示せば、次のようになる。すなわち、スレッド処理多重度管理テーブル２２及びトレース条件管理テーブル３２の内容が図６に示すような状態にある場合に、たとえば識別子が”４００”であるスレッドの登録依頼があったとき、そのスレッドで実行されるメソッドのクラス／メソッド”ｃｌａｓｓ　Ａ：：Ｍｅｔｈｏｄ　Ｙ”はトレース条件管理テーブル３２中に存在し、そのトレース・レベルは”４”である。また、そのレコード中には前記特定関係にある他のメソッド”ｃｌａｓｓ　Ｂ：：Ｍｅｔｈｏｄ　Ｚ”が存在する。そして、この特定関係にあるメソッドはスレッド処理多重度管理テーブル２２において識別子が”２００”のスレッドに係るものとして既に登録されている。したがって、図７に示されるとおり、登録依頼された識別子”４００”のスレッドがスレッド処理多重度管理テーブル２２にエントリされるとともに、識別子”２００”及び”４００”の各レコードのトレース・レベルが”４”に設定される。

【００５２】

その後、識別子が”４００”のスレッドについてトレース処理の可否の問合せがトレース実行部２３（図２）からあったとすれば、スレッド処理多重度管理テーブル２２中の当該レコードにおけるトレース・レベル”４”がトレース実行部２３に通知される。トレース実行部２３はこの通知されたトレース・レベルに応じてトレース処理を実行し、又は一部省略する。

【００５３】

一方、アプリケーション実行制御部３から、識別子が”２００”のスレッドについて登録抹消依頼があったとすれば、そのスレッドに係るクラス／メソッド”ｃｌａｓｓ　Ｂ：：Ｍｅｔｈｏｄ　Ｚ”はトレース条件管理テーブル３２においてトレース・パターン２のレコード中に存在する。また同じレコード中に、特定関係にあるメソッド／クラス”ｃｌａｓｓ　Ａ：：Ｍｅｔｈｏｄ　Ｙ”も定義されている。したがってこの場合、トレース制御部３１は、スレッド処理多重度管理テーブル２２における識別子”２００”のレコードを削除するとともに、メソッドが特定関係にある識別子”４００”のレコードにおけるトレース・レベルを”０”に設定する。以後、識別子”４００”のスレッドにおけるトレース処理は禁止されることになる。

【００５４】

図１０は本発明の第３の実施形態に係るサーバにおけるシステム構成及び処理シーケンスを示す。この実施形態では、サーバ・プログラムが共有ライブラリ中のメソッドを呼び出すことによってマルチスレッドによる処理が行われる場合について示している。図１０のシステムはサーバ・プログラム１１１、サーバ・プログラム１１１によって使用される共有ライブラリ１１２、実行中のスレッドのみを登録し、この登録情報に基づいてトレース処理の実行可否の問合せに応答するトレース制御部１１３、共有ライブラリ中の実行中のメソッドから依頼されるトレース処理の実行要求に応じ、トレース処理の実行可否をトレース制御部１１３に問い合わせ、その応答に基づいてトレース処理を実行し又は省略するトレース実行部１１４、及びスレッドの登録がなされるスレッド処理多重度管理テーブル１１５を備える。スレッド処理多重度管理テーブル１１５の各レコードには、登録されるスレッドの識別子、スレッドが実行するメソッドのメソッド名、及びトレース・レベルが記録される。トレース・レベルが”０”であれば対応スレッドにおけるトレース処理は禁止され、”５”であれば許容される。

【００５５】

この構成において、サーバ・プログラム１１１が共有ライブラリ１１２中のメソッドたとえば”ＳＱＬＥｘｅｃ（）”を呼び出すと（処理①）、メソッドを実行

するスレッドが生成される。すると、メソッドはまず、トレース制御部１１３に対し、生成されたスレッドの登録を依頼する（処理②）。この依頼は、図１１の例ではトレース制御部１１３中の”ａｄｄＴｈｒｅａｄＩｎｆｏ（）”を呼び出すことによって行っている。

【００５６】

この依頼を受けると、トレース制御部１１３は、依頼されたスレッドの登録を行う（addThreadInfo()；処理③）。このときトレース制御部１１３は、スレッド処理多重度管理テーブル１１５において同一メソッドを実行しているスレッドが登録されているかどうかを調べ、登録されていなければトレース・レベルを”０”とし、登録されていれば、双方のトレース・レベルを”５”に設定する（処理④）。図１０の例では、登録されるスレッド”５００”が実行しているメソッド”ＳＱＬＥｘｅｃ”は、スレッド”１００”においても実行されているため、双方のトレース・レベルは”５”に設定されている。

【００５７】

共有ライブラリ１１２中の実行中のメソッドが、トレース処理たとえば”Ｔｒａｃｅ”Ｔｒａｃｅ２””の実行をトレース実行部１１４に依頼すると（処理⑤）、トレース実行部１１４は、メソッドを実行しているスレッドの識別子を特定して、トレース制御部１１３に対し、トレース処理の実行可否を問い合わせる（処理⑥）。この問合せに応じ、トレース制御部１１３は、問合せに係るスレッドについて、スレッド処理多重度管理テーブル１１５におけるトレース・レベルに基づきトレース処理の実行可否を判定し（処理⑦）、その結果をトレース実行部に通知する（処理⑥）。図１０の例ではこの処理をメソッド”ｃｈｅｃｋＴｒａｃｅ（）”によって行っている。たとえば問合せに係るメソッドが”ＳＱＬＥｘｅｃ”であれば、トレース・レベルが”５”であるため、トレース処理の実行を許容する旨を通知する。この通知に基づいてトレース実行制御部１１４は、依頼されたトレース処理の実行を、ログ・ファイル２４を出力先として行う（setLogEvent()；処理⑧）。

【００５８】

共有ライブラリ１１２中の実行中のメソッドが終了するとき、メソッドはそれ

を実行しているスレッドの登録抹消をトレース制御部１１３に依頼する（Call removeThreadInfo()；処理⑨）。この依頼に応じてトレース制御部１１３はそのスレッドの登録を抹消する（removeThreadInfo()；処理(10)）。その際、削除するレコードとメソッド名が同一であるレコードについては、さらにメソッド名が同一である他のレコードが存在しない限り、トレース・レベルを”０”とする。たとえば、図１０のスレッド処理多重度管理テーブル１１５の状態においてスレッド識別子が”１００”であるレコードが削除されるとき、そのメソッド名”ＳＱＬＥｘｅｃ”が同一であるスレッド識別子が”５００”のレコードのトレース・レベルは”０”に変更される。

【００５９】

本実施形態におけるトレース制御部１１３の処理は、メソッドの識別をメソッド名のみによって行っている以外は、第１実施形態におけるトレース制御部２１の場合とほぼ同じである。本実施形態によっても第１実施形態の場合と同様の効果を得ることができる。

【００６０】

なお、本発明は上述実施形態に限定されることなく適宜変形して実施することができる。たとえば、上述においてはスレッド処理多重度管理テーブル２２においてスレッド識別子の項目が設けられているが、この項目は、スレッド毎にメソッドが登録されていれば、設けなくてもよい。ただし、テーブルのサーチ処理が煩雑になる可能性がある。また、上述においては言及しなかったが、トレース条件管理テーブル３２の内容を設定し又は更新するための入力を受け入れ、その設定又は更新を行う手段を設けるようにしてもよい。

【００６１】

【発明の効果】

以上説明したように本発明によれば、起動中のスレッド毎に、実行中のルーチンの登録を行い、この登録情報に基づいて、実行中のルーチンにおけるトレース処理のレベルを判定するようにしたため、プログラムの実行環境に大きな影響を与えることなく、共有リソースを同時にアクセスすることによって生じる問題の解決に必要な情報の収集を行うことができる。

【図面の簡単な説明】

【図１】

本発明を適用し得るＯＲＢ通信基盤を基礎とするクライアント・サーバ・システムを示す図である。

【図２】

本発明の一実施形態に係るＯＲＢ通信基盤を基礎とするサーバにおけるシステム及び該システムにおける処理シーケンスを示す図である。

【図３】

図２のシステムにおける登録処理を示すフローチャートである。

【図４】

図２のシステムにおける登録抹消処理を示すフローチャートである。

【図５】

図２のシステムにおける実行可否チェック処理を示すフローチャートである。

【図６】

本発明の第２の実施形態に係るシステム及び処理の流れを示す図である。

【図７】

図６のシステムのトレース制御部による登録処理を示すフローチャートである。

【図８】

図６のシステムのトレース制御部による登録抹消処理を示すフローチャートである。

【図９】

図６のシステムのトレース制御部による実行可否チェック処理を示すフローチャートである。

【図１０】

本発明の第３の実施形態に係るサーバにおけるシステム構成及び処理シーケンスを示す。

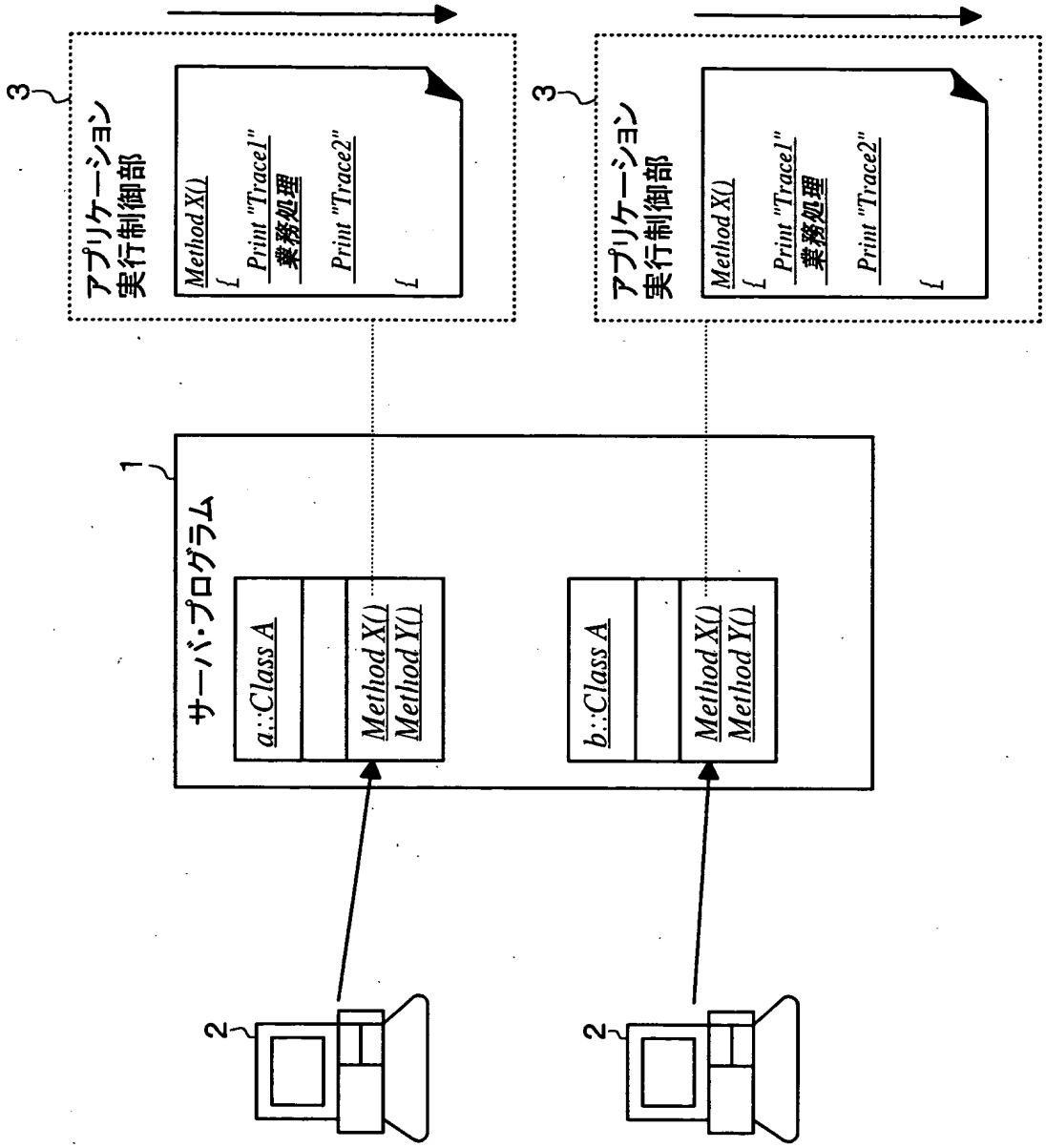
【符号の説明】

１，１１１：サーバ・プログラム、２：クライアント・マシーン、３：アプリ

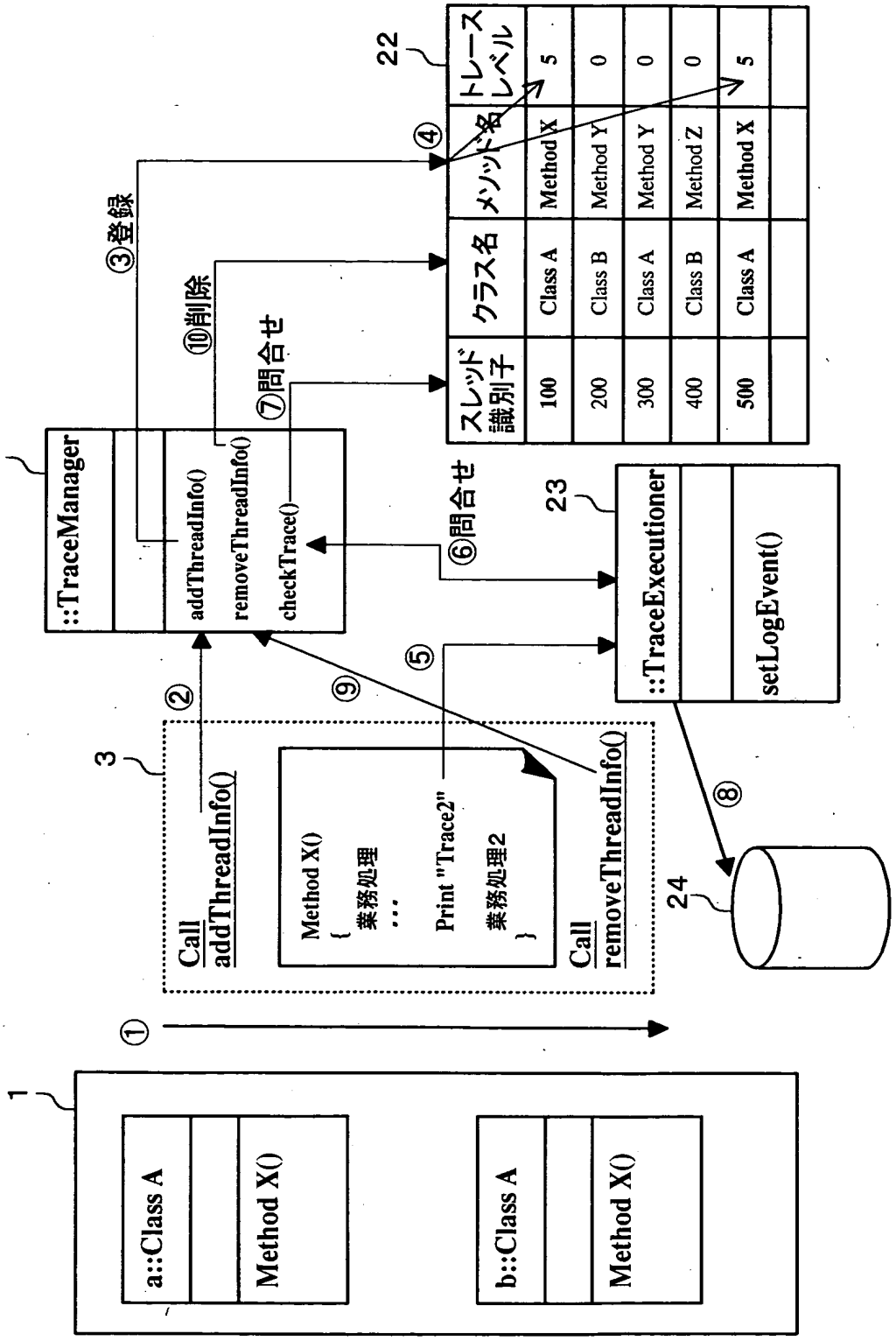
ケーション実行制御部、２１，３１，１１３：トレース制御部、２２，１１５：スレッド処理多重度管理テーブル、２３：トレース実行部、２４：ログ・ファイル、３２：トレース条件管理テーブル、１１２：共有ライブラリ。

【書類名】　　図面

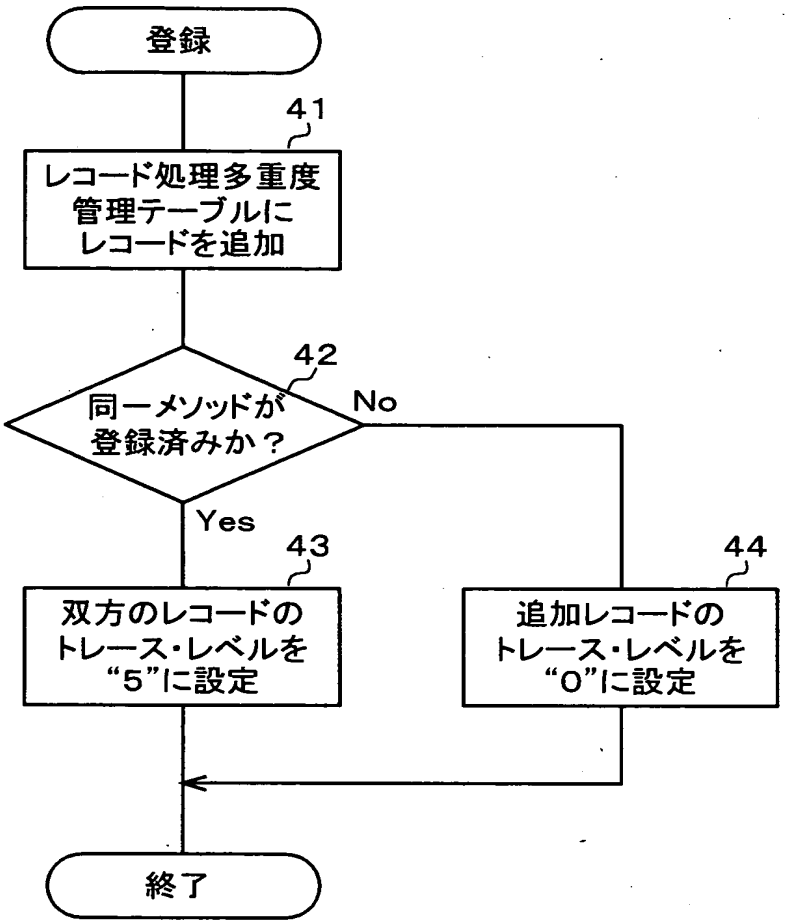
【図１】


3
アプリケーション
実行制御部
Method X()
{
Print "Trace1"
業務処理
Print "Trace2"
}
3
アプリケーション
実行制御部
Method X()
{
Print "Trace1"
業務処理
Print "Trace2"
}
1
サーバ・プログラム
a::Class A
Method X()
Method Y()
b::Class A
Method X()
Method Y()
2
2

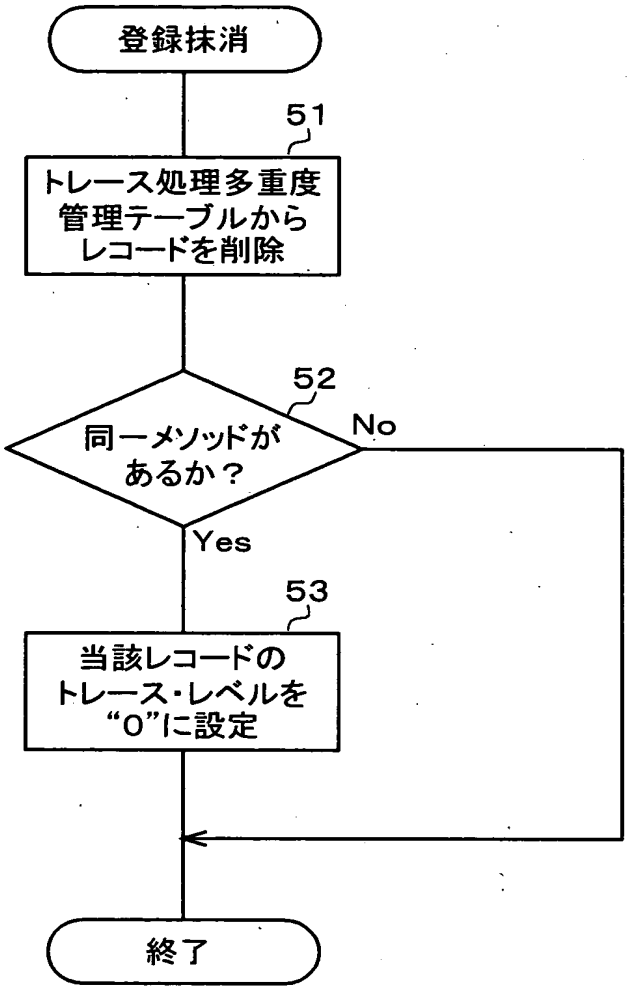
【図２】


1
a::Class A
Method X()
b::Class A
Method X()
①
3
Call addThreadInfo()
Method X()
{
業務処理
…
Print "Trace2"
業務処理2
}
Call removeThreadInfo()
②
21
::TraceManager
addThreadInfo()
removeThreadInfo()
checkTrace()
③登録
④
22
スレッド識別子
クラス名
メソッド名
トレースレベル
100 Class A Method X 5
200 Class B Method Y 0
300 Class A Method Y 0
400 Class B Method Z 0
500 Class A Method X 5
⑤
⑥問合せ
⑦問合せ
⑧
⑨
⑩削除
23
::TraceExecutioner
setLogEvent()
24

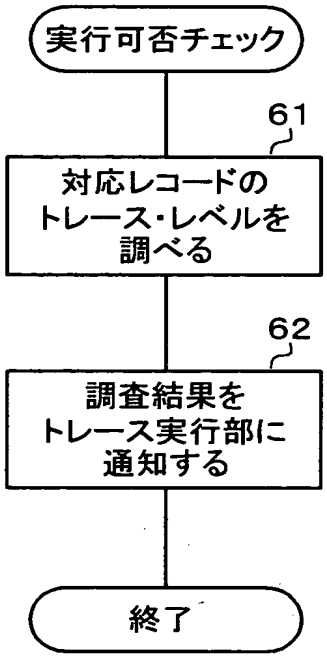
【図３】


登録
41
レコード処理多重度
管理テーブルに
レコードを追加
42
同一メソッドが
登録済みか？
No
Yes
43
双方のレコードの
トレース・レベルを
“5”に設定
44
追加レコードの
トレース・レベルを
“0”に設定
終了

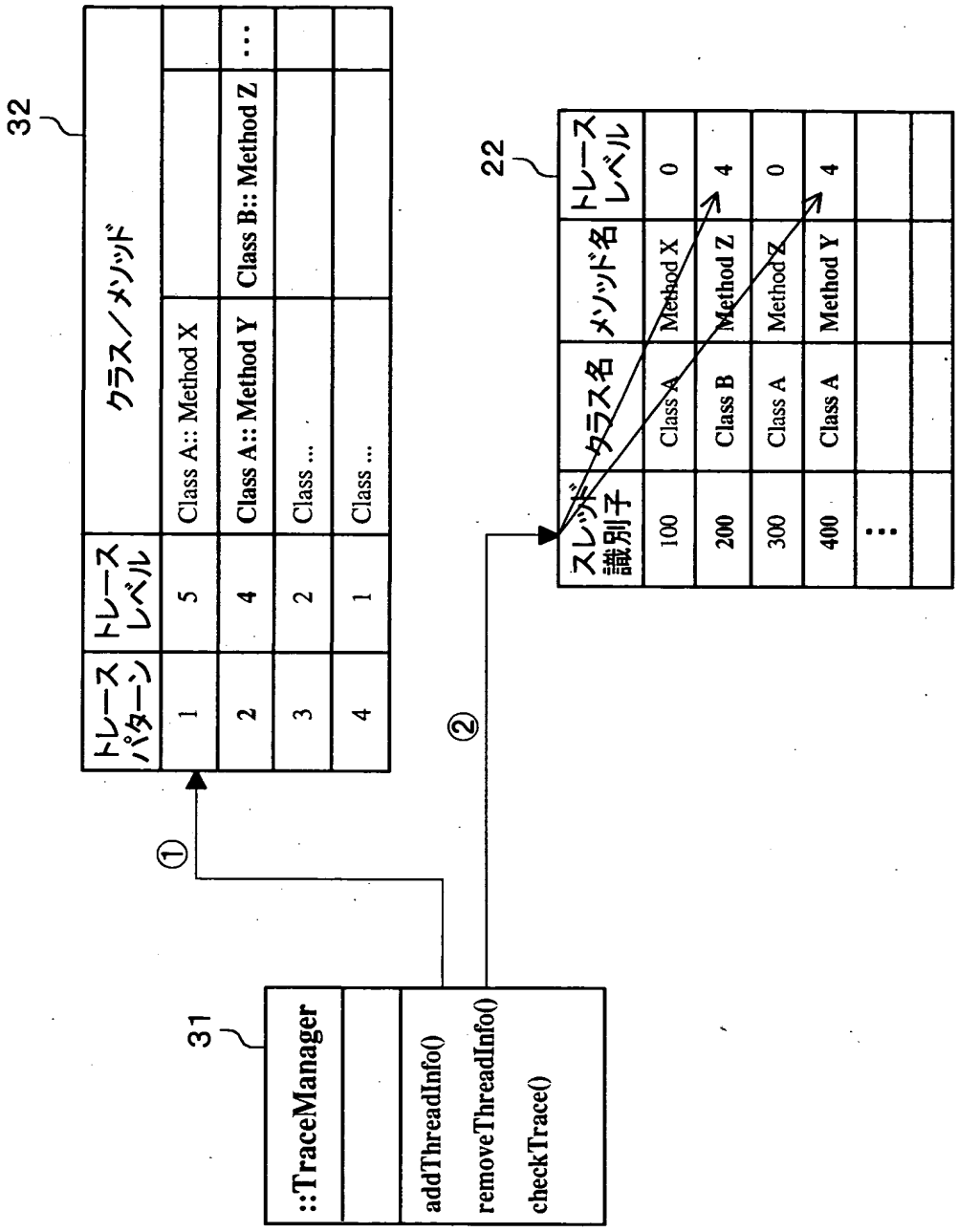
【図４】


登録抹消
51
トレース処理多重度
管理テーブルから
レコードを削除
52
同一メソッドが
あるか？
No
Yes
53
当該レコードの
トレース・レベルを
“0”に設定
終了

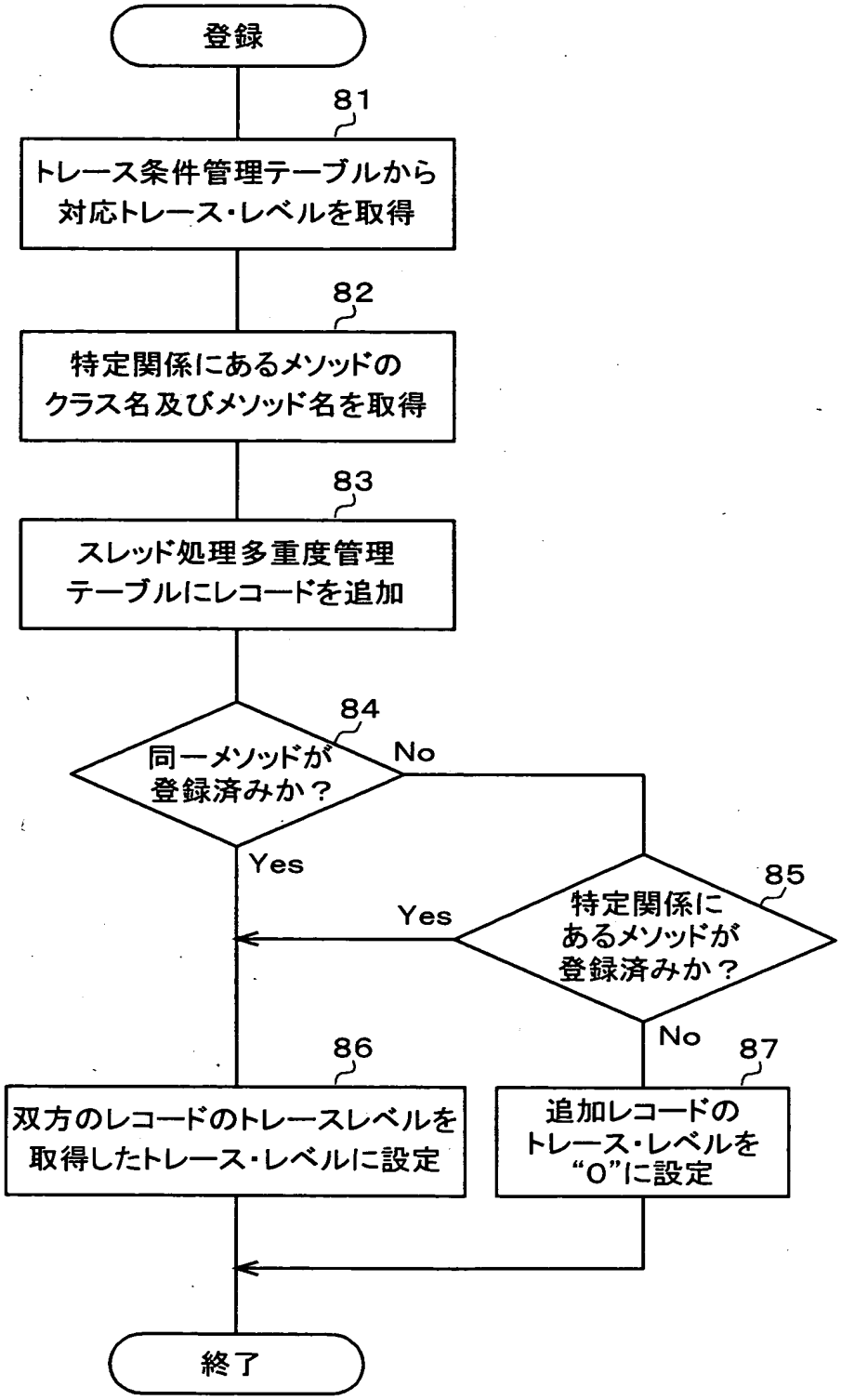
【図５】



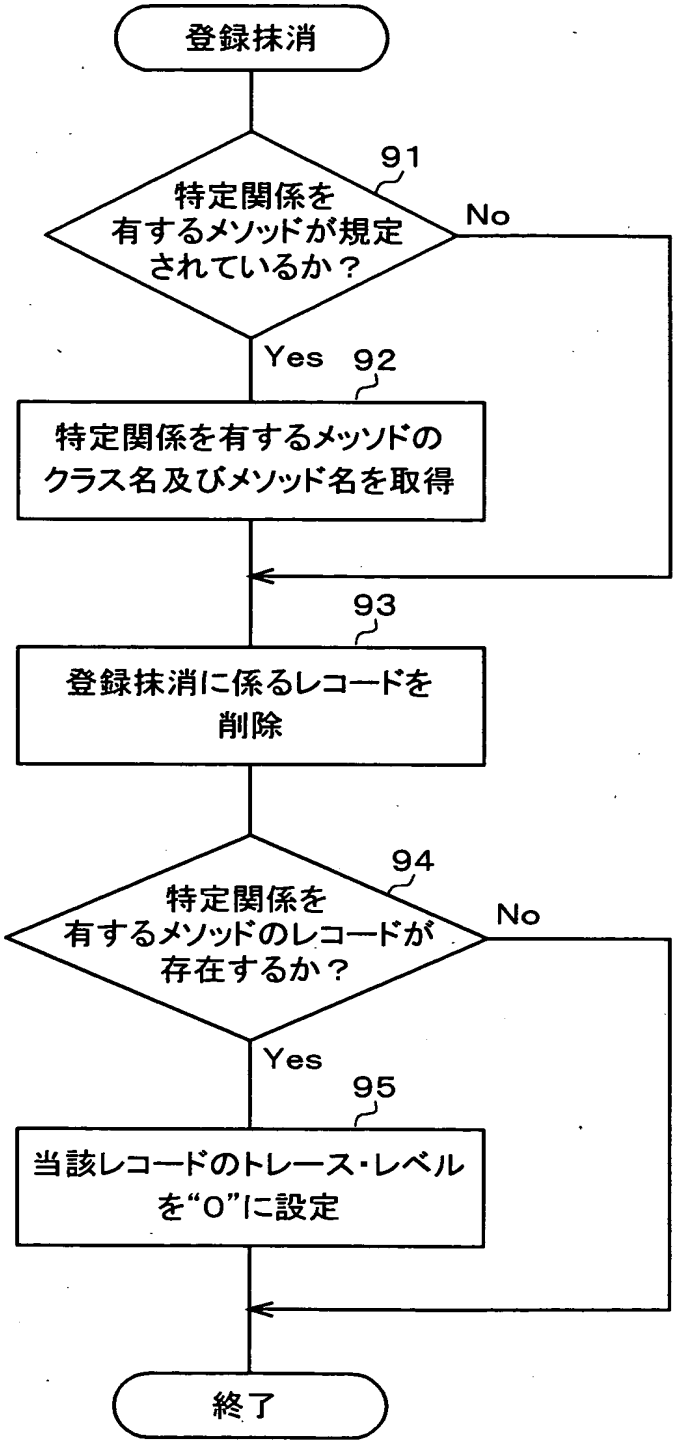
【図６】


32
トレースパターン
トレースレベル
クラス／メソッド
1
5
Class A:: Method X
2
4
Class A:: Method Y
3
2
Class ...
4
1
Class ...
Class B:: Method Z
...
22
スレッド識別子
クラス名
メソッド名
トレースレベル
100
Class A
Method X
0
200
Class B
Method Z
4
300
Class A
Method Z
0
400
Class A
Method Y
4
①
②
31
::TraceManager
addThreadInfo()
removeThreadInfo()
checkTrace()

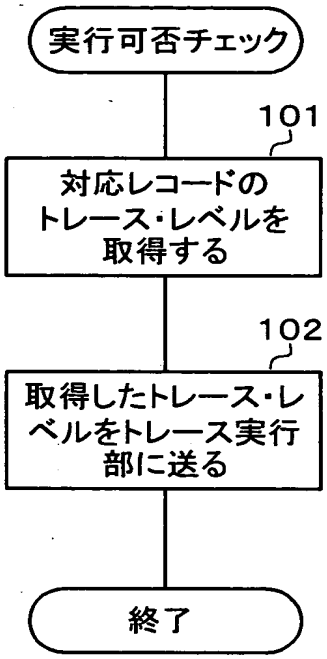
【図７】


登録
81
トレース条件管理テーブルから
対応トレース・レベルを取得
82
特定関係にあるメソッドの
クラス名及びメソッド名を取得
83
スレッド処理多重度管理
テーブルにレコードを追加
84
同一メソッドが
登録済みか？
No
Yes
85
特定関係に
あるメソッドが
登録済みか？
Yes
No
86
双方のレコードのトレースレベルを
取得したトレース・レベルに設定
87
追加レコードの
トレース・レベルを
“0”に設定
終了

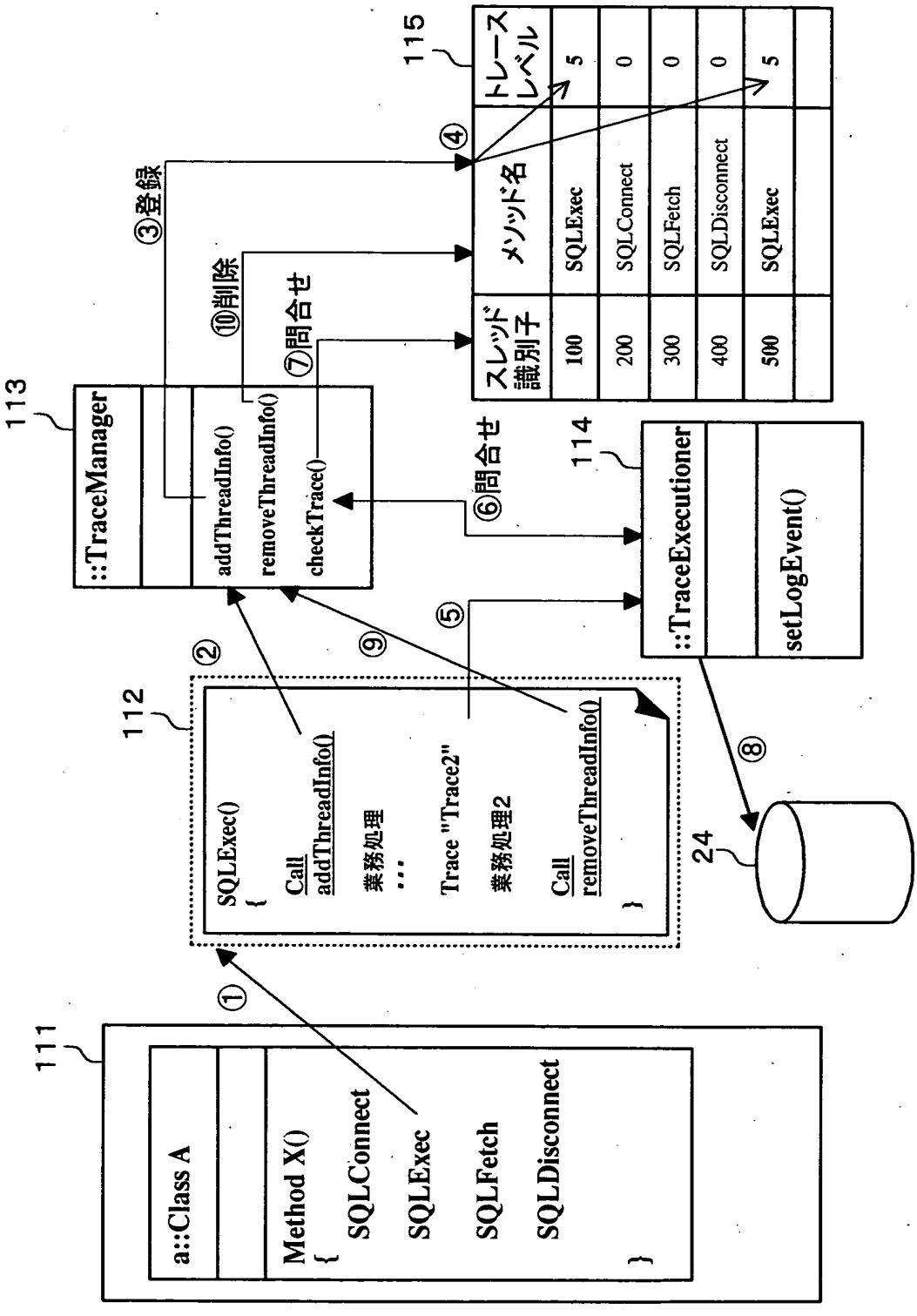
【図８】


登録抹消
91
特定関係を
有するメソッドが規定
されているか？
No
Yes
92
特定関係を有するメッソドの
クラス名及びメソッド名を取得
93
登録抹消に係るレコードを
削除
94
特定関係を
有するメソッドのレコードが
存在するか？
No
Yes
95
当該レコードのトレース・レベル
を“0”に設定
終了

【図９】


実行可否チェック
101
対応レコードの
トレース・レベルを
取得する
102
取得したトレース・レ
ベルをトレース実行
部に送る
終了

【図１０】


111
a::Class A
Method X()
{
SQLConnect
SQLExec
SQLFetch
SQLDisconnect
}
①
②
112
SQLExec()
{
Call addThreadInfo()
業務処理
…
Trace "Trace2"
業務処理2
Call removeThreadInfo()
}
113
::TraceManager
addThreadInfo()
removeThreadInfo()
checkTrace()
③登録
④
⑤
⑥問合せ
⑦問合せ
⑧
⑨
⑩削除
114
::TraceExecutioner
setLogEvent()
115
スレッド識別子
メソッド名
トレースレベル
100 SQLExec 5
200 SQLConnect 0
300 SQLFetch 0
400 SQLDisconnect 0
500 SQLExec 5
24

【書類名】　　要約書

【要約】

【課題】　プログラムの実行環境に大きな影響を与えることなく、共有リソースを同時にアクセスすることによって生じる問題の解決に必要な情報の収集を行うことができるようにする。

【解決手段】　処理状況に関する情報を収集するトレース処理が含まれるルーチンをマルチスレッドで実行可能な情報処理装置において、起動中のスレッド毎に実行中のルーチンの登録を行い（ステップ３１～３３、３５）、実行中のルーチンにおけるトレース処理のレベルを登録情報に基づいて判定する（ステップ３３、３６）。実行中のルーチンにおけるトレース処理のレベル判定は、そのルーチンと同一又は所定の関係にあるルーチンの登録がなされているか否かに基づいて行うことができる。

【選択図】　　図３

認定・付加情報

特許出願の番号	特願２００２－３６６３４５
受付番号	５０２０１９１５９６３
書類名	特許願
担当官	末武　実　　１９１２
作成日	平成１５年　３月２４日

＜認定情報・付加情報＞

【提出日】　平成14年12月18日

【特許出願人】

【識別番号】　390009531

【住所又は居所】　アメリカ合衆国１０５０４、ニューヨーク州　アーモンク　ニュー　オーチャード　ロード

【氏名又は名称】　インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】

【識別番号】　100086243

【住所又は居所】　神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　大和事業所内

【氏名又は名称】　坂口　博

【代理人】

【識別番号】　100091568

【住所又は居所】　神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　大和事業所内

【氏名又は名称】　市位　嘉宏

【代理人】

【識別番号】　100108501

【住所又は居所】　神奈川県大和市下鶴間１６２３番１４　日本アイ・ビー・エム株式会社　知的所有権

【氏名又は名称】　上野　剛史

【復代理人】　申請人

【識別番号】　100085408

【住所又は居所】　東京都中央区日本橋２丁目１番１号　櫻正宗ビル９階

【氏名又は名称】　山崎　隆

次頁無

出　願　人　履　歴　情　報

識別番号　　〔３９０００９５３１〕

１．変更年月日　　２００２年　６月　３日
［変更理由］　住所変更
住　所　　アメリカ合衆国１０５０４、ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
氏　名　　インターナショナル・ビジネス・マシーンズ・コーポレーション